EXHIBIT D

US 20020080871A1

(19) United States
(12) Patent Application Publication
Fallon et al.
(10) Pub. No.: US 2002/0080871 A1
(43) Pub. Date: Jun. 27, 2002

(54) SYSTEM AND METHOD FOR DATA FEED ACCELERATION AND ENCRYPTION

(75) Inventors: **James J. Fallon**, Armonk, NY (US); **Paul F. Pickel**, Bethpage, NY (US); **Stephen J. McErlain**, New York, NY (US)

Correspondence Address:
Frank V. DeRosa
F. CHAU & ASSOCIATES, LLP
Suite 501
1900 Hempstead Turnpike
East Meadow, NY 11554 (US)

(73) Assignee: **Realtime Data, LLC**

(21) Appl. No.: **09/969,987**

(22) Filed: **Oct. 3, 2001**

Related U.S. Application Data

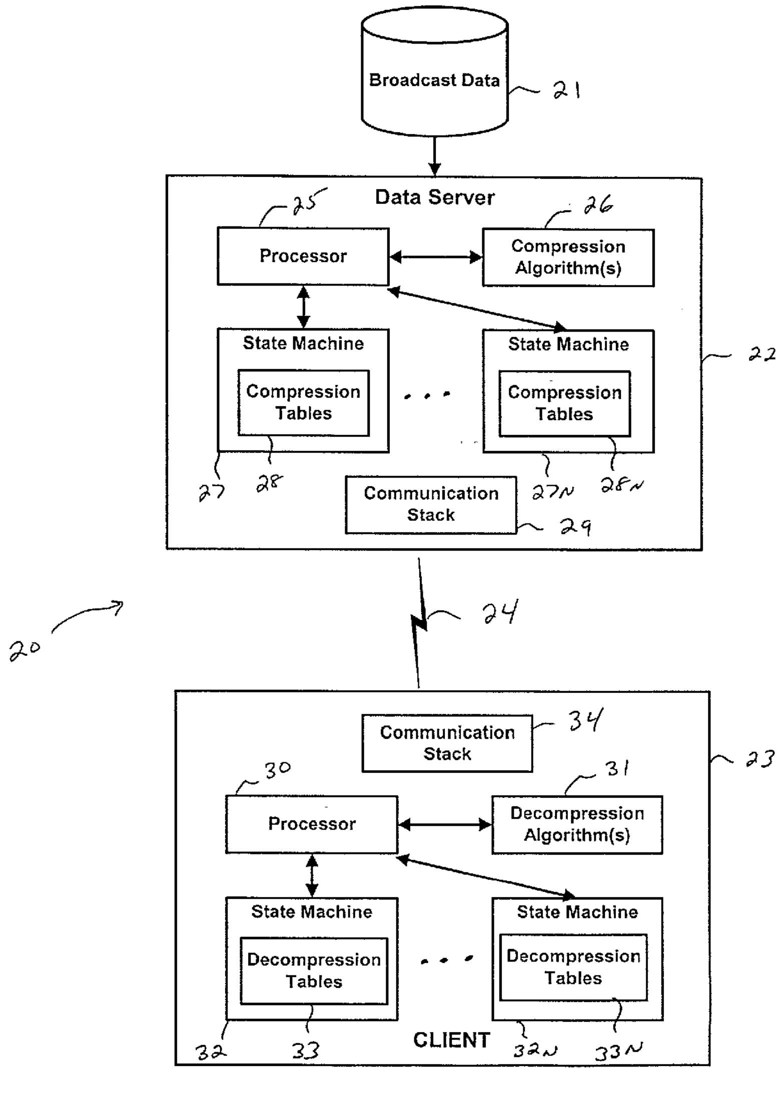
(63) Non-provisional of provisional application No. 60/237,571, filed on Oct. 3, 2000.

Publication Classification

(51) Int. Cl.[7] **H04B 1/66**
(52) U.S. Cl. **375/240**

(57) **ABSTRACT**

Systems and methods for providing accelerated transmission of broadcast data, such as financial data and news feeds, over a communication channel using data compression and decompression to provide secure transmission and transparent multiplication of communication bandwidth, as well as reduce the latency associated with data transmission of conventional systems.



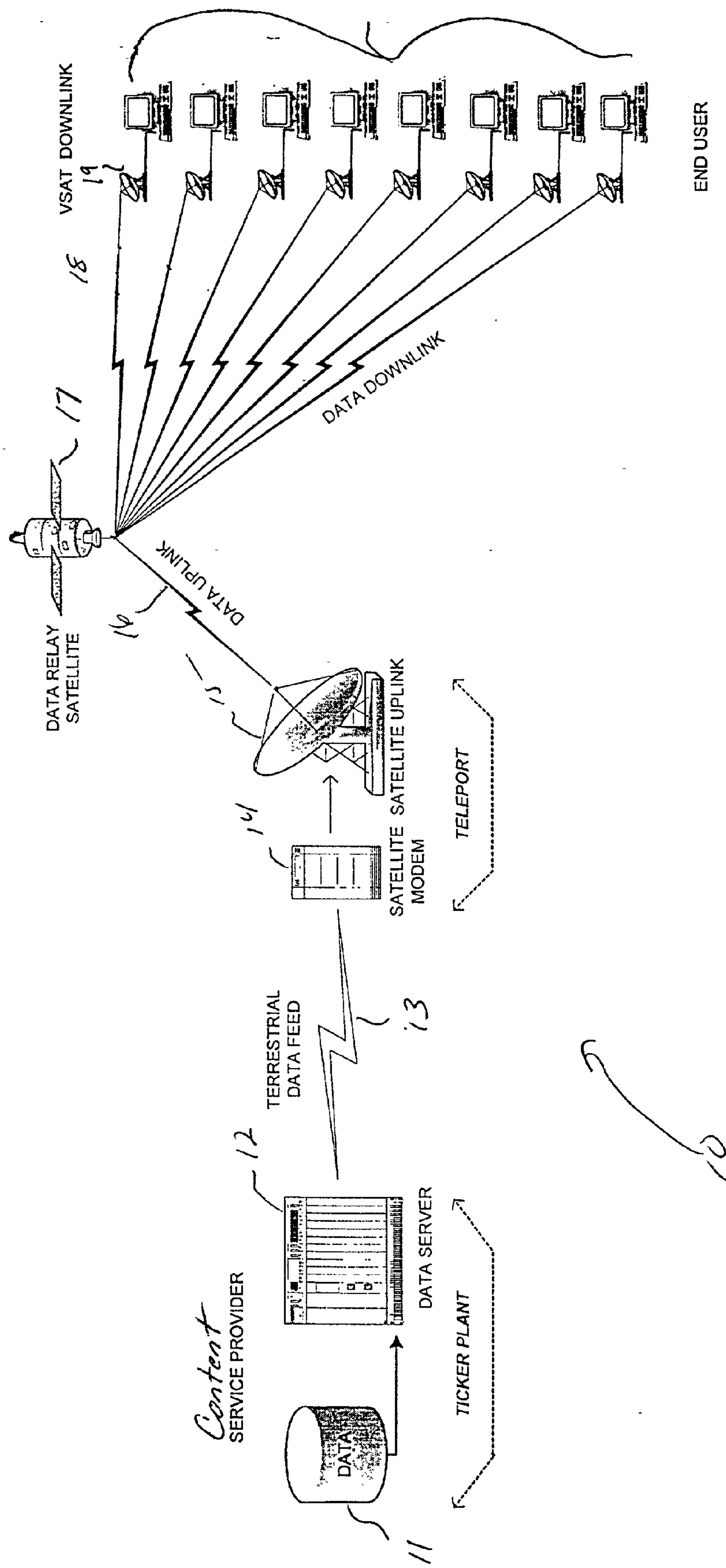

Content SERVICE PROVIDER
DATA
11
12
DATA SERVER
TICKER PLANT
TERRESTRIAL DATA FEED
13
14
SATELLITE MODEM
SATELLITE UPLINK
15
TELEPORT
DATA UPLINK
16
DATA RELAY SATELLITE
17
DATA DOWNLINK
18
VSAT DOWNLINK
19
END USER
10

Fig. 1

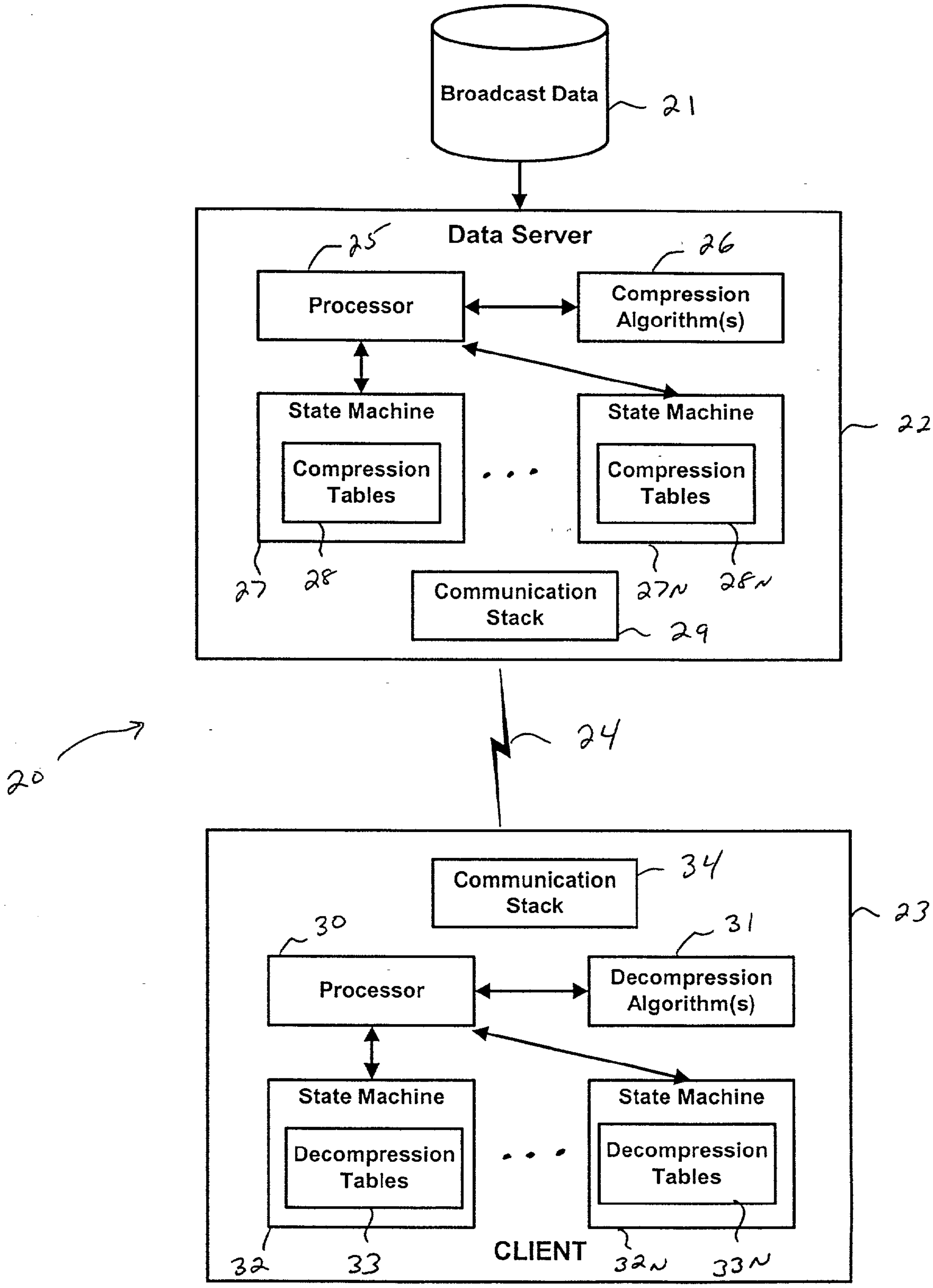

Broadcast Data
21
Data Server
25
26
Processor
Compression Algorithm(s)
State Machine
Compression Tables
State Machine
Compression Tables
22
27
28
27N
28N
Communication Stack
29
24
20
Communication Stack
34
23
30
31
Processor
Decompression Algorithm(s)
State Machine
Decompression Tables
State Machine
Decompression Tables
CLIENT
32
33
32N
33N

Fig. 2

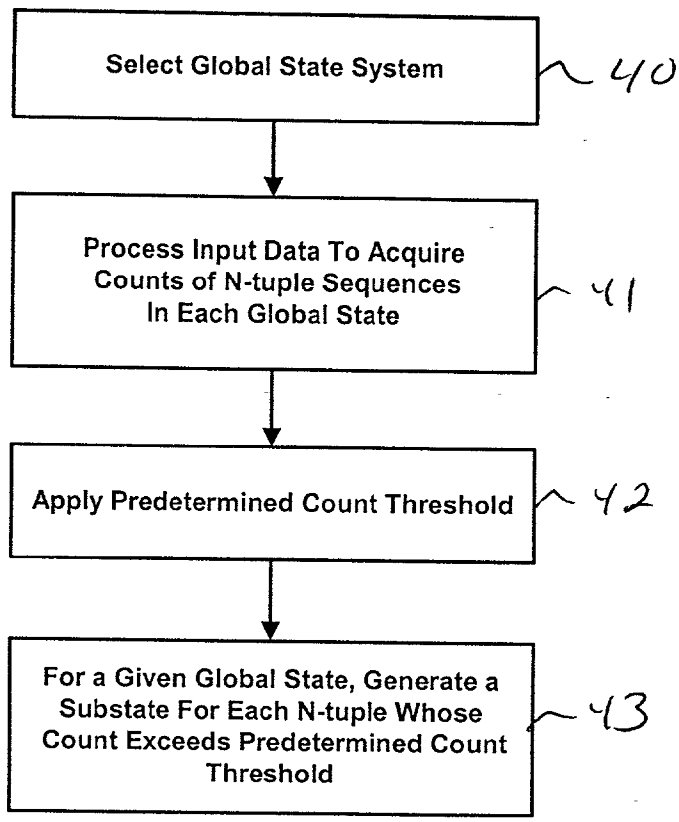
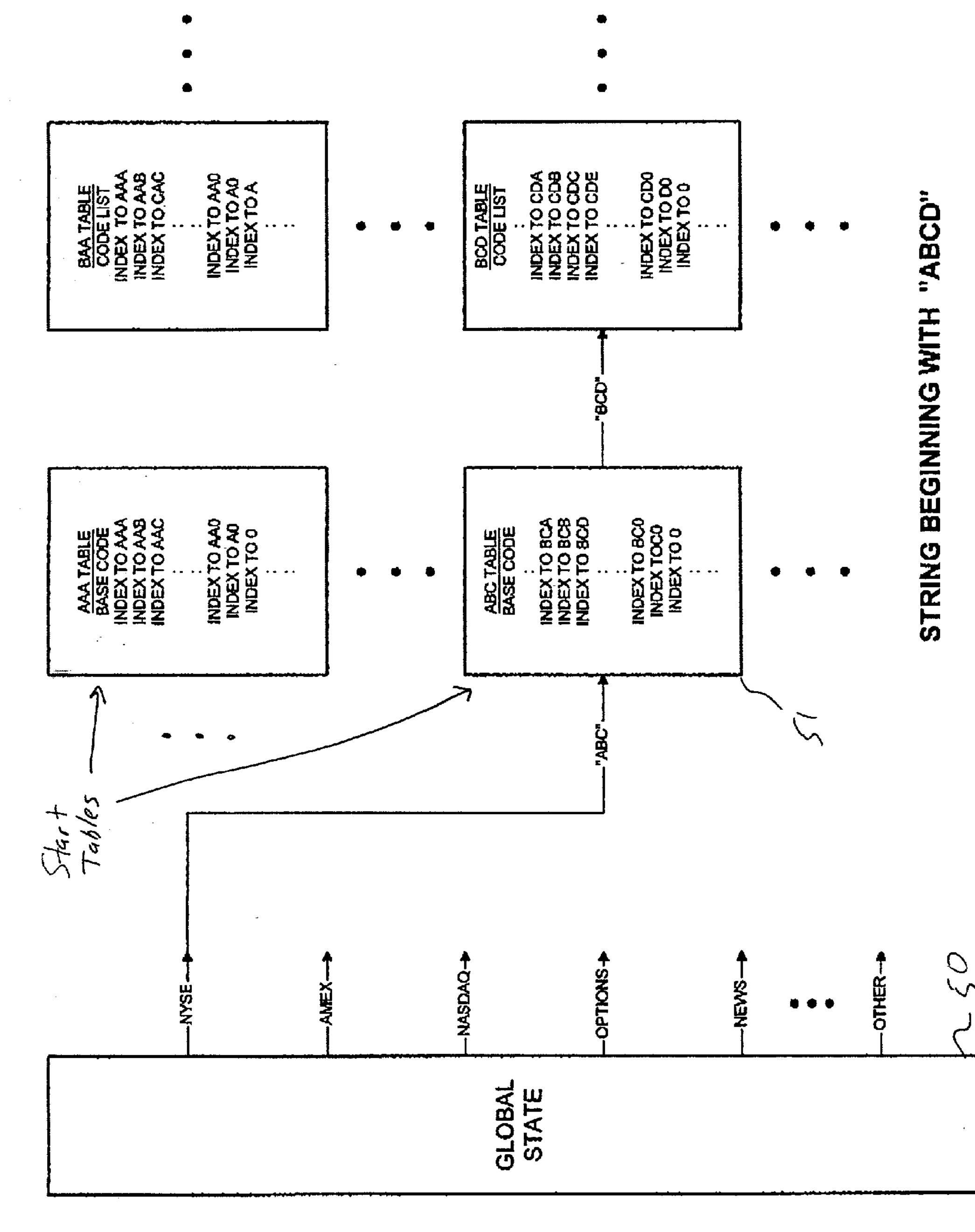

Select Global State System
40
Process Input Data To Acquire Counts of N-tuple Sequences In Each Global State
41
Apply Predetermined Count Threshold
42
For a Given Global State, Generate a Substate For Each N-tuple Whose Count Exceeds Predetermined Count Threshold
43

Fig. 3



STRING BEGINNING WITH "ABCD"

Fig. 4

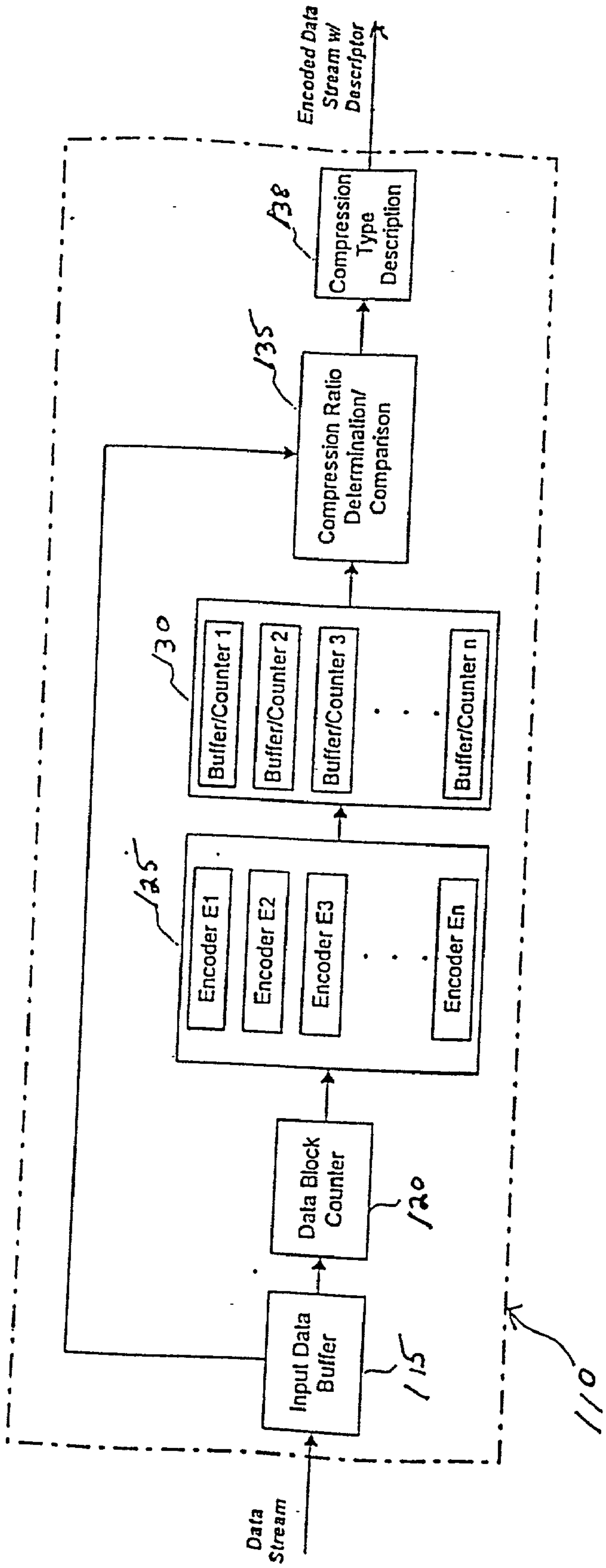

Data Stream
Input Data Buffer
115
Data Block Counter
120
Encoder E1
Encoder E2
Encoder E3
Encoder En
125
Buffer/Counter 1
Buffer/Counter 2
Buffer/Counter 3
Buffer/Counter n
130
Compression Ratio Determination/ Comparison
135
Compression Type Description
138
Encoded Data Stream w/ Descriptor
110

FIGURE 5

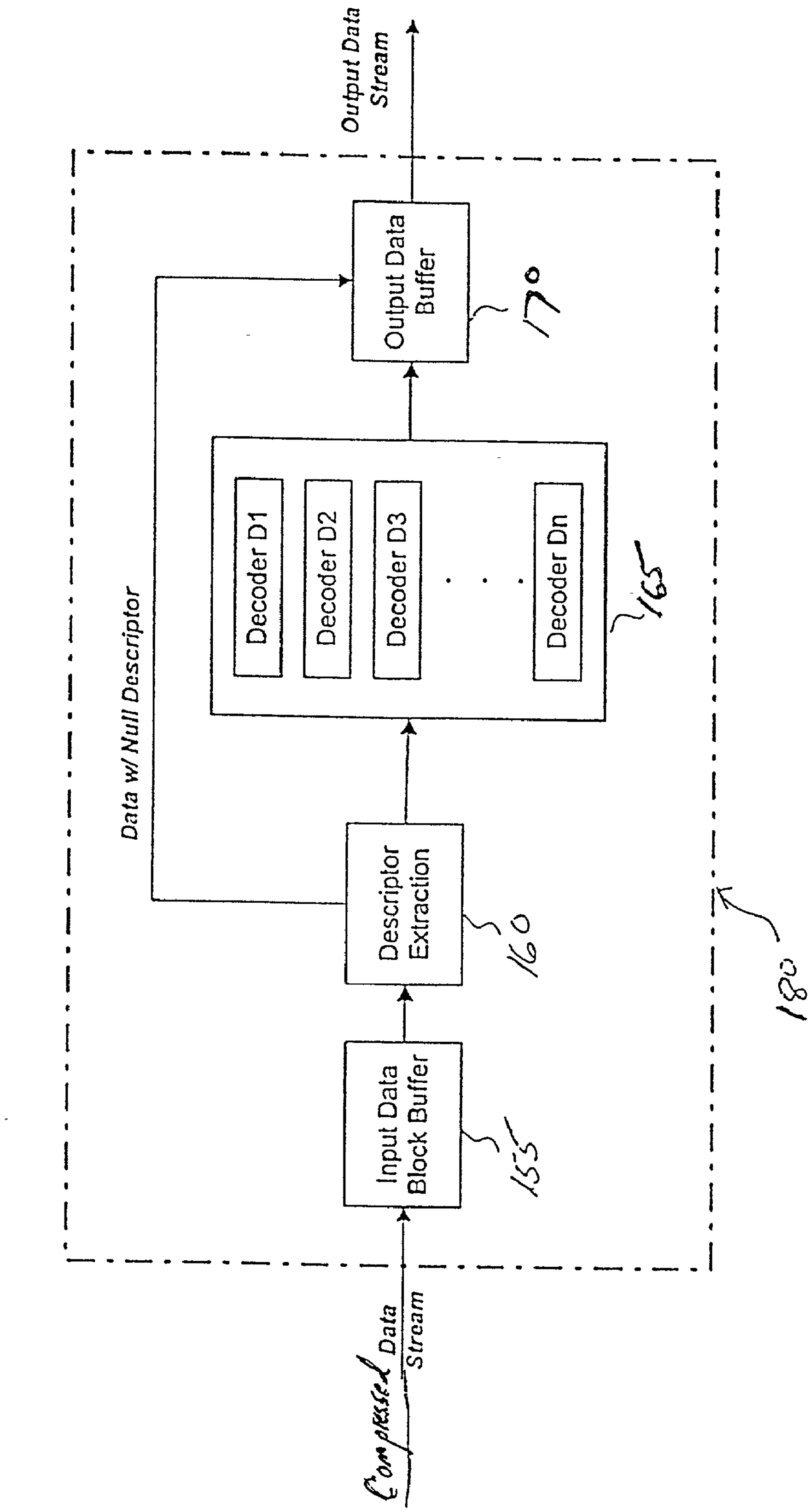

Output Data Stream
Output Data Buffer
170
Data w/ Null Descriptor
Decoder D1
Decoder D2
Decoder D3
. . .
Decoder Dn
165
Descriptor Extraction
160
Input Data Block Buffer
155
180
Compressed Data Stream

FIG. 6

SYSTEM AND METHOD FOR DATA FEED ACCELERATION AND ENCRYPTION

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application is based on a United States provisional application Serial No. 60/237,571, filed on Oct. 3, 2000, which is fully incorporated herein by reference.

BACKGROUND

[0002] 1. Technical Field

[0003] The present invention relates generally to systems and method for providing data transmission, and in particular, to systems and method for providing accelerated transmission of data, such as financial data and news feeds, over a communication channel using data compression and decompression to provide secure data transfer and effectively increase the bandwidth of the communication channel and/or reduce the latency of data transmission.

[0004] 2. Description of Related Art:

[0005] The financial information services industry covers a broad range of financial information ranging from basic stock quotations to analyst reports to detailed pricing of callable bonds. Users of financial information can generally be divided into two segments—Information users and Analytics users. Information users range from nonfinance business professionals to curious stock market investors and tend to seek basic financial information and data. Analytical users on the other hand, tend to be finance professionals who require more arcane financial information and utilize sophisticated analytical tools to manipulate and analyze data (e.g. for writing option contracts).

[0006] Historically, proprietary systems, such as Bloomberg, Reuters and Bridge Information, have been the primary electronic source for financial information to both the informational and analytical users. These closed systems required dedicated telecommunications lines and product-specific hardware. The most typical installations are land-based networking solutions such as T1 or ISDN, and satellite-based "wireless" solutions at speeds of 384 kbps.

[0007] Latency of financial data is and news is critical to the execution of financial transactions. Indeed the more timely receipt of financial data from various sources including the New York Stock Exchange, American Stock Exchange, National Association of Securities Dealers (NASDAQ), Options Exchange, Commodities Exchanges, and Futures presents a fundamental advantage to those who trade. Latency is induced by the long time taken to compress and encrypt data prior to transmission, along with the associated time to decrypt and decompress. Often current methods of encryption and compression take as much or substantially more time than the actual time to transmit the uncompressed, unencrypted data. Thus one problem within the current art is the latency induced by the act of encryption, compression, decryption, and decompression.

[0008] Modern data compression algorithms suffer from poor compression, high latency, or both. Within the present art algorithms such as Lempel-Ziv, modified/embellished Lempel-Ziv, Binary Arithmetic, and Huffman coding are essentially generic algorithm having a varied effectiveness on different data types. Also small increases in compression to the negentropy limit of the data generally require exponentially greater periods of time and substantially higher latency. Generic algorithms are currently utilized as data types and content format is constantly changed within the financial industry. Many changes are gradual however there are also abrupt changes, such as the recent switch to decimalization that has imposed substantial requirements on data transmission bandwidth infrastructure within the financial industry. Thus another problem within the current art is the high latency and poor compression due to the use of generic data compression algorithms on financial data and news feeds.

[0009] Within the financial and news feeds, data is often segregated into packets for transmission. Further, in inquiry response type systems, as found in many financial research systems, the size of request packets and also response packets is quite small. As such, response servers often wait for long periods of time (for example 500 msec) to aggregate data packets prior to transmission back to the inquirer. By aggregating the data and then applying compression, higher compression ratios are often achieved. This then translates to lower data communications costs or more customers served for a given amount of available communications bandwidth. Thus another problem within the current art is the substantial latency caused by aggregating data packets due to poor data compression efficiency and packet overhead.

[0010] Thus, given the importance of receiving financial information over computer networks, an improved system and method for providing secure point-to-point solution for transparent multiplication of bandwidth over conventional communication channels is highly desirable.

[0011] These and other limitations within the current art are solved by the present invention.

SUMMARY OF THE INVENTION

[0012] The present invention is directed to systems and methods for providing accelerated transmission of data, such as financial data and news feeds, over a communication channel using data compression and decompression to provide secure data transfer and effectively increase the bandwidth of the communication channel. The present invention is preferably applied in broadcast-based systems and inquiry response-based systems in which the data structure of the.

[0013] In one aspect of the present invention, a method for providing accelerated transmission of data over a network comprises the steps of:

[0014] receiving a data stream for transmission over a communication channel;

[0015] compressing, in real-time, the data stream at a compression rate that increases the effective bandwidth of the communication channel;

[0016] transmitting the compressed data stream over the communication channel; and

[0017] decompressing, in real time, the compressed data stream received over the communication channel.

[0018] In another aspect, the step of compressing comprises the steps of compressing the data stream using a

statistical compression algorithm, wherein the statistical compression algorithm uses code tables that are derived based on a data model associated with the data stream. The statistical compression algorithm preferably comprises Huffman encoding or Arithmetic encoding.

[0019] In yet another aspect, the code tables for the Huffman or Arithmetic encoding comprise a state machine. The state machine comprises one or more global states each having one or more code tables associated therewith. The complexity of the code tables provides a virtual encryption and, thus a level of security that obviates the need to use additional encryption on the data stream.

[0020] In another aspect of the present invention, a method for providing accelerated transmission of data over a network comprises the steps of:

[0021] receiving a data stream for transmission over a communication channel;

[0022] compressing, in real-time, the data stream;

[0023] transmitting the compressed data stream over the communication channel; and

[0024] decompressing, in real time, the compressed data stream received over the communication channel, wherein a latency associated with the compressing, transmitting and decompression steps is less than a latency associated with transmitting the data stream in uncompressed format.

[0025] In yet another aspect of the present invention, a method for providing accelerated transmission of data over a network comprises the steps of:

[0026] compressing a data stream using a content-dependent compression system that is constructed based on a data model associated with the data stream;

[0027] transmitting the compressed data stream over a communication channel; and

[0028] utilizing a content-independent compression system to compress the data stream, when a compression ratio obtained using the content-dependent system falls below a predetermined threshold, or when a latency associated with the data transmission exceeds a predetermined threshold.

[0029] These and other aspects, features and advantages, of the present invention will become apparent from the following detailed description of preferred embodiments that is to be read in connection with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0030] **FIG. 1** is a block diagram of a system in which the present invention may be implemented for transmitting broadcast data;

[0031] **FIG. 2** is a block diagram of a system and method for providing accelerated transmission of data over a communication channel according to an embodiment of the present invention;

[0032] **FIG. 3** is a flow diagram illustrating a method for generating compression/decompression state machines according to one aspect of the present invention;

[0033] **FIG. 4** is a diagram illustrating an exemplary encoding table structure used for compressing/decompressing data according to the present invention, which may be generated using the process of **FIG. 3**.

[0034] **FIG. 5** is a diagram of a system/method for providing content independent data compression, which may be implemented for providing accelerated data transmission according to the present invention; and

[0035] **FIG. 6** is a diagram of a system/method for providing content independent data decompression, which may be implemented for providing accelerated data transmission according to the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0036] The present invention is directed to systems and methods for providing accelerated transmission of broadcast data, such as financial data and news feeds, over a communication channel using data compression and decompression to provide secure transmission and transparent multiplication of communication bandwidth, as well as reduce the latency associated with data transmission of conventional systems.

[0037] In general, the term "accelerated" data transmission refers to a process of receiving a data stream for transmission over a communication channel, compressing the broadcast data stream in real-time (wherein the term "real time" as used herein collectively refers to substantially real time, or at real time, or greater than real time) at a compression rate that increases the effective bandwidth of the communication channel, and transmitting the compressed broadcast data over the communication channel. The effective increase in bandwidth and reduction of latency of the communication channel is achieved by virtue of the fast than real-time, real-time, near real time, compression of a received data stream prior to transmission.

[0038] For instance, assume that the communication channel has a bandwidth of "B" megabytes per second. If a data transmission controller is capable of compressing an input data stream with an average compression rate of 3:1, then data can be transmitted over the communication channel at an effective rate of up to 3*B megabytes per second, thereby effectively increasing the bandwidth of the communication channel by a factor of three.

[0039] Further, when the receiver is capable decompressing (in substantially real time, real time, or faster than real time) the compressed data stream at a rate approximately equal to the compression rate, the point-to-point transmission rate between the transmitter and receiver is transparently increased. Advantageously, accelerated data transmission can mitigate the traditional bottleneck associated with, e.g., local and network data transmission.

[0040] Additionally, if the compression and decompression are accomplished in real-time or faster, the compressed, transmitted and decompressed data is available before the receipt of an equivalent uncompressed stream. The latency of data transmission over the communication channel is achieved when the total time for compression, transmission, and decompression, is less than the total time for transmitting the data in uncompressed form.

[0041] It is to be understood that the present invention may be implemented in various forms of hardware, software, firmware, or a combination thereof. Preferably, the present invention is implemented on a computer platform including hardware such as one or more central processing units (CPU) or digital signal processors (DSP), a random access memory (RAM), and input/output (I/O) interface(s). The computer platform may also include an operating system, microinstruction code, and dedicated processing hardware utilizing combinatorial logic or finite state machines. The various processes and functions described herein may be either part of the hardware, microinstruction code or application programs that are executed via the operating system, or any combination thereof.

[0042] It is to be further understood that, because some of the constituent system components described herein are preferably implemented as software modules, the actual system connections shown in the Figures may differ depending upon the manner in that the systems are programmed. General purpose computers, servers, workstations, personal digital assistants, special purpose microprocessors, dedicated hardware, or and combination thereof may be employed to implement the present invention. Given the teachings herein, one of ordinary skill in the related art will be able to contemplate these and similar implementations or configurations of the present invention.

[0043] **FIG. 1** is a diagram illustrating a system in which the present invention may be implemented. The system **10** comprises content **11** and data server **12** associated with a service provider of broadcast data. The content **11** comprises information that is processed by the data server **12** to generate a broadcast, e.g., a news feed or financial data feed. As explained in further detail below, the data server **12** employs data compression to encode/encrypt the broadcast data **11** prior to transmission over various communication channels to one or more client site systems of subscribing users, which comprise the necessary software and hardware to decode/decrypt the compressed broadcast data in real-time. In the exemplary embodiment of **FIG. 1**, the communication channels comprise a landline **13** that feeds the compressed broadcast data to a satellite system comprising modem **14** and an uplink system **15**, which provides a data uplink **16** to a relay **17**. The relay **17** provides data downlinks **18** to one or more downlink systems **19**.

[0044] Advantageously, the proprietary software used by the data server **12** to compress the data stream in real-time and software used by the workstations **19** to decompress the data stream in real-time effectively provides a seamless and transparent increase in the transmission bandwidth of the various communication channels used, without requiring modification of existing network infrastructure.

[0045] Referring now to **FIG. 2, a** block diagram illustrates a system/method for providing accelerated transmission of data according to one embodiment of the present invention. More specifically, **FIG. 2** illustrates embodiments of a broadcast data server (transmitter) and client system (receiver) for implementing accelerated transmission and real-time processing of broadcast data. Broadcast data **21** (comprising one or more different broadcast types) is processed by data server **22** prior to transmission to client **23** over a communication channel **24**. The data server **22** utilizes a processor **25** (e.g., microprocessor, digital signal processor, etc.) for executing one or more compression algorithms **26** for compressing (in real-time) the broadcast data **21** prior to transmission. In preferred embodiments, compression is achieved using Huffman or Arithmetic encoding, wherein one or more state machines **27**-**27***n* are constructed based on a-priori knowledge of the structure and content of one or more given broadcast and data feeds.

[0046] As explained in further detail below, each state machine **27**-**27***n* comprises a set of compression tables that comprise information for encoding the next character (text, integer, etc.) or sequence of characters in the broadcast data feed, as well as pointers which point to the next state (encoding table) based on the character or character sequence. As explained in greater detail below, a skeleton for each state machine **27**-**27***n* (nodes and pointers) is preferably built by finding sequences of characters (n-tuples) that frequently appear in a given data input. Once a skeleton has been determined, a large set of data is processed through the system and counts are kept of character n-tuples for each state. These counts are then used to construct the compression tables associated with the state machine to provide statistical compression. The compressed data is transmitted over the communication channel **24** via a communication stack using any suitable protocol (e.g., RTP (real time protocol) using RTCP (real-time control protocol), TCP/IP, UDP, or any real-time streaming protocol with suitable control mechanism).

[0047] Similarly, the client **23** comprises a processor **30** for executing one or more decompression algorithms **31**. Depending one the data feed type, one of a plurality of decompression state machines **32**-**32***n* are used to decompress the compressed data stream received by the client **23** via communication stack **34**. Each state machine **32**-**32***n* comprises a set of decompression tables **33**-**33***n* that comprise information for decode the next encoded character (or symbol) or sequence of symbols in the compressed broadcast data feed, as well as pointers which point to the next state based on the symbol or symbol sequence. For each compression state machine **27**-**27***n* in the data server, a corresponding decompression state machine **32**-**32***n* is needed in the client **23** to decompress the associated data stream.

[0048] Advantageously, a compression/decompression scheme according to the present invention using Huffman or Arithmetic encoding provides secure transmission via de facto or virtual "encryption" in a real-time environment. Indeed, virtual encryption is achieved by virtue of the fast, yet complex, data compression using Huffman tree, for example, without necessarily requiring actual encryption of the compressed data and decryption of the compressed data. Because of the time-sensitive nature of the market data, and the ever-changing and data-dependent nature of the arithmetic scheme, decryption is virtually impractical, or so complex and useless as to render the data worthless upon eventual decoding.

[0049] However, data compression using Huffman or Arithmetic encoding yields encoded data that is very difficult to decode than current encryption schemes such as plain text or simple bit shuffling codes as currently used by broadcast service providers. An attacker must have the compression model and the tables used to compress the data stream to be able to obtain useful information from it. Thus,

at one level of security, the client-side decompression tables are preferably stored in encrypted form and are decrypted on being loaded into the processor **30** (e.g., general purpose processor, DSP, etc.) using an encryption/decryption key that is validated for a subscribing user. In this manner, a client will be unable to use the tables on other processors or sites or after terminating a service contract.

[**0050**] Since Huffman compression uses the same bit code for a character each time it appears in a given context, an attacker with a very large data set of compressed and uncompressed data could possibly reconstruct the tables, assuming the overall model were known. Arithmetic compression, on the other hand, generates different bit patterns for the same character in the same context depending on surrounding characters. Arithmetic encoding provides at least an order of magnitude more difficult to recover the tables from the compressed and uncompressed data streams.

[**0051**] The following is a detailed discussion of a compression scheme using Huffman or Arithmetic encoding for providing accelerated transmission of broadcast data according to one aspect of the present invention. It is to be appreciated that the present invention is applicable with any data stream whose statistical regularity may be captured and represented in a state machine model. For example, the present invention applies to packetized data streams, in which the packets are limited in type format and content.

[**0052**] In one embodiment using Huffman or Arithmetic encoding, each character or character sequence is encoded (converted to a binary code) based on the frequency of character or character sequence in a given "context". For a given context, frequently appearing characters are encoded with few bits while infrequently appearing characters are encoded with more bits. High compression ratios are obtained if the frequency distribution of characters in most contexts is highly skewed with few frequently appearing characters and many characters seldomly (or never) appear.

[**0053**] Referring now to **FIG. 3, a** flow diagram illustrates a method for generating compression/decompression state machines according to one aspect of the present invention. The "context" in which a character (or character sequence) is encoded in a given broadcast stream is based on a "global state" that represents packet type and large-scale structure and the previous few characters. The first step in building a compression scheme involves selecting a global state system based on the packet structure of the broadcast model (step **40**). More specifically, a global state system is constructed based on a priori knowledge of the data stream model, e.g., the packet type frequency and structure of the broadcast model. By way of example, one model for financial data may comprise four global states representing: a beginning of packet, an options packet, a NYSE (New York Stock Exchange) packet and some other packet type. Further, additional codes may be added to the encoding tables to indicate global state transitions (e.g., for an end of packet code in the broadcast model). If there is internal structure to packets, such as a header with different statistics than the body, additional global states could be added.

[**0054**] Once a global state system is selected, training samples from an associated data stream are passed through the global model to acquire counts of frequencies of the occurrence of n-tuple character sequences ending in each of the model states (step **41**). In a preferred embodiment, the n-tuples comprise character sequences having **1**, **2** and **3** characters. Using the acquired counts, sub-states (or "local states") of the predefined global states are constructed based on previous characters in the data stream. A local state may depend on either none, 1, 2, or 3 (or more) previous characters in the stream. To provide a practical limitation, a predetermined count threshold is preferably applied to the count data (step **42**) and only those sequences that occur more often than the count threshold are added as local states (step **43**). For example, if a three-character sequence does not occur sufficiently frequently, the count for the last two characters is tested, etc.

[**0055**] It is to be understood that any character sequence length "n" may be implemented depending on the application. The longer the allowed character sequence, the more memory is needed to store the encoding tables and/or the lower the count threshold should be set.

[**0056**] As samples of the data are passed through the state model, character (and transition code) counts for each context are accumulated. These counts are used to build the Huffman or Arithmetic coding tables. The construction of the global and local models is an iterative process. The count threshold for forming local states can be adjusted depending on the application. For instance, a larger threshold will result in less local states but less compression as well. Further, a comparison of statistics in local or global states may suggest adding or deleting global states.

[**0057**] The construction of the global model requires knowledge of the data stream packet structure. The construction of the local states is automatic (once the threshold is set).

[**0058**] **FIG. 4** is a diagram of an exemplary encoding table structure according to one aspect of the present invention. As indicated above, a global state **50** is determined from a field within the data stream to be encoded. Various forms of data have different identifiers, for example one type of NYSE data may be identified by 0x00, AMEX 0x01, NASDAQ 0x02, Options Data 0x03, News 0x04, and so forth. The first characters (binary pattern) within the string are utilized as a direct index to the NYSE start table **51**. For example, if we are using three character long strings and the characters are "ABC", the "ABC" table is used. In general all three-character combinations will have start tables, however it is possible that certain strings occur so infrequently as to not warrant tables and may be transmitted in an uncompressed format. It should be mentioned that the tables may use any tuple of characters, three being a convenient value for present processor, memory, compression, and latency considerations.

[**0059**] The next defining three-bit sequence is used as an index from the "ABC" table to the next table. For example for the string "ABCD" the next sequence is "BCD". It should be noted that all tables are not complete, i.e. only frequently occurring values above a threshold are employed to limit memory usage and achieve an acceptable compression level. Also it should be noted that if the whole string was simply "ABC" a base code value within the start table is used **51**.

[**0060**] This process is repeated to the depth of the code tables, i.e. maximum string length matches for compression. An n-tuple indexing system according to the present inven-

tion is highly effective for hashing and indexing, optimizing the overall performance of the compression system. It should be noted that this system of organization works equally well for compression systems that concatenate codes from multiple tables or have unique codes per fixed string.

[0061] As noted above with reference to **FIGS. 1 and 2**, a compression scheme according to the present invention may be implemented in any system to provide accelerated data transmission to multiple client site systems. Preferably, the client site systems may connect at any time, so minimal immediate history may be used (since a newly connected site must be able to pick up quickly). A system according to an embodiment of the present invention uses statistical compression (Huffman or Arithmetic coding) using fixed (or adaptive) tables based on the statistics of a data feed sample. As noted above, it has been determined that the statistical compression schemes described herein are well adapted for use with structured data streams having repetitive data content (e.g., stock symbols and quotes, etc.) to provide fast and efficient data compression/decompression.

[0062] The following discussion provides further details regarding the preparation of statistical-based encoding tables and their use for compression/decompression according to the present invention. During a data compression process, the selection of which encoding table to use for compression is preferably based on up to n (where n is preferably equal to 3) preceding characters of the message. In an exemplary broadcast model tested by the present inventors, a data stream comprises messages that begin with an ID code in the range 0-31 with the remainder of the message being characters in the range 32-127. It was found that approximately half of the messages in a given sample began with ID code 0x0c and half of the remainder began with ID code 0x0f. Thus, a separate encoding table is preferably used for a message ID code. Further, separate table sets are used for messages beginning with 0x0c and with 0x0f, with the remaining messages lumped together in another table.

[0063] Each table has an additional termination code. The termination code in a "start table" indicates the end of a compression block. The termination code in all other tables indicates the end of the message. Thus, the start table comprises **33** entries and all other tables have **97** entries.

[0064] Using one table for each 3-character context would require prohibitive amounts of memory. For example, a complete one-character context would require 33+3*97=324 tables. Then, a complete two-character context would require 324*97=31,428 tables. And finally, a complete three-character context would require 324*97*97=3,048,516 tables. Preferably, as described above, the application of a count threshold at each context size reduces the amount of tables. Only when a context occurs at greater than the threshold rate in the sample will a table be created for that context.

[0065] Each table entry includes a link to the next table to be used. For instance, in an "abc" context table, the entry for next character "d" would point to the "bcd" table, if such table was created. If such table was not created, the entry for next character "d" would point to the "cd" table, if such table existed. If no "cd" table exists, the "d" table would be used and if that fails, a base table for the message type would be used.

[0066] For a client site system to pick up the broadcast feed at any time, clearly identifiable synchronization points are preferably included in the compressed data stream. In a preferred embodiment, data is compressed in blocks with each block comprising some number of complete messages. Preferably, each compressed block ends with at least four bytes with each bit being logic 1 and no interior point in the compressed block will comprise 32 consecutive 1 bits. The compressed block preferably begins with two bytes giving the decompressed size of the block shifted to guarantee that the first byte of the compressed block is not all 1's. Thus, to achieve synchronization, the client site system can scan the input compressed data stream for 4 bytes of 0xff, wherein the next byte not equal to 0xff is deemed the start of a compressed block. In other words, the receiver will accumulate the compressed data until at least a sequence of 4 bytes each having a value of 0xff is detected in the input stream, at which point decompression will commence on the compressed input stream.

[0067] In another embodiment of the present invention, if a compressed block is more than 6 bytes longer than the uncompressed data, the data block is transmitted uncompressed preceded by the shifted two-byte count with the high bit set and trailed by 4 bytes of 0xff.

[0068] The following is discussion of a method for preparing Huffman Tables according to one aspect of the present invention. The Huffman codes generated by a conventional optimal algorithm have been modified in various ways in accordance with the present invention. First, in order that there not be 32 consecutive one bits in the data stream except at the end of a compression block, a termination code in each table comprises all 1 bits.

[0069] Further, to reduce space required for decompression tables, and ensure no sequence of 32 1 bits, each code is preferably decoded as follows:

[0070] a) The first 7 bits are used to index into a table. If the character code is no more than 7 bits, it can be read directly;

[0071] b) otherwise, some number N of initial bits is discarded and the next 7 bits are used to index a second table to find the character.

[0072] Based on these steps, preferably, no character code can use more than 14 bits and all codes of more than 7 bits must fit into the code space of the N initial bits. If N is 3, for instance, then no code can use more than 10 bits. To achieve this, the code space required for all optimal codes of more than 7 bits is first determined, following by a determining the initial offset N. Every code comprising more than N+7 bits is preferably shortened, and other codes are lengthened to balance the code tree. It is possible that this may cause the code space for codes over 7 bits to increase so that N may need to be decreased. Preferably, this process is performed in a manner that causes minimal reduction in the efficiency of the codes.

[0073] The above modifications to convention optimal algorithm yields codes in which no non-termination code ends in more than 7 1 bits, no non-termination code begins with more than 6 1 bits, no termination code is more than 14 1 bits and no non-termination packet start code begins with more than 5 1 bits. Thus, in the middle of a packet, a

sequence of no more than 13 bits of logic 1 can occur, while, at the end of a packet, a sequence of no more than 26 bits of logic 1 can occur.

[0074] In another embodiment of the present invention, Arithmetic compression can be used instead of Huffman encoding. The tables for Arithmetic encoding are preferably constructed such that a sequence of 32 bits of logic 1 will not occur in the interior of a message (which is important for a random sign-on in the middle of the stream).

[0075] Arithmetic compression provides an advantage of about 6% better compression than Huffman and uses half as much memory for tables, which allows the number of tables to be increased). Indeed, the addition of more tables and/or another level of tables yields more efficient compression. Although Arithmetic compression may take about 6 times as long as Huffman, this can certainly be improved by flattening the subroutine call tree (wherein there is a subroutine call for each output bit.) In summary, a compression scheme according to one aspect of the invention utilizes a state machine, wherein in each state, there is a compression/decompression table comprising information on how to encode/decode the next character, as well as pointers that indicated which state to go to based on that character. A skeleton of the state machine (nodes and pointers) is preferably built by finding sequences of characters that appear often in the input. Once the skeleton has been determined, a large set of data is run through the system and counts are kept of characters seen in each state. These counts are then used to construct the encode/decode tables for the statistical compression.

[0076] Other approaches may be used to build the skeleton of the state machine. A very large fraction of the traffic on a certain feed consists of messages in the digital data feed format, which is fairly constrained. It may be possible to build by hand a skeleton that takes into account this format. For instance, capital letters only appear in the symbol name at the beginning. This long-range context information can be represented with our current approach. Once a basic skeleton is in place, the structure could be extended for sequences that occur frequently.

[0077] The above-described statistical compression schemes provide content-dependent compression and decompression. In other words, for a given data stream, the above schemes are preferably structured based on the data model associated with the given stream. It is to be appreciated, however, that other compression schemes may be employed for providing accelerated data transmission in accordance with the present invention for providing effectively increased communication bandwidth and/or reduction in latency. For instance, in another embodiment of the present invention, the data compression/decompression techniques disclosed in U.S. Pat. No. 6,195,024, issued to J. Fallon on Feb. 27, 2001, entitled "Content Independent Data Compression Method and System" may be used in addition to, or in lieu of, the statistical based compression schemes described above.

[0078] In general, a content-independent data compression system is a data compression system that provides an optimal compression ratio for an encoded stream regardless of the data content of the input data stream. A content-independent data compression method generally comprises the steps of compressing an input data stream, which comprises a plurality of disparate data types, using a plurality of different encoders. In other words, each encoder compresses the input data stream and outputs blocks of compressed data. An encoded data stream is then generated by selectively combining compressed data blocks output from the encoders based on compression ratios obtained by the encoders.

[0079] Because a multitude of different data types may be present within a given input data stream, or data block, to it is often difficult and/or impractical to predict the level of compression that will be achieved by any one encoding technique. Indeed, rather than having to first identify the different data types (e.g., ASCII, image data, multimedia data, signed and unsigned integers, pointers, etc.) comprising an input data stream and selecting a data encoding technique that yields the highest compression ratio for each of the identified data types, content-independent data compression advantageously applies the input data stream to each of a plurality of different encoders to, in effect, generate a plurality of encoded data streams. The plurality of encoders are preferably selected based on their ability to effectively encode different types of input data. Ultimately, the final compressed data stream is generated by selectively combining blocks of the compressed streams output from the plurality of encoders. Thus, the resulting compressed output stream will achieve the greatest possible compression, regardless of the data content.

[0080] In accordance with another embodiment of the present invention, a compression system may employ both a content-dependent scheme and a content-independent scheme, whereby the content-dependent scheme is used as the primary compression/decompression system and the content-independent scheme is used in place of, or in conjunction with, the content dependent scheme, when periodically checked "compression factor" meets a predetermined threshold. For instance, the compression factor may comprise a compression ratio, wherein the compression scheme will be modified when the compression ration falls below a certain threshold. Further, the "compression factor" may comprise the latency of data transmission, wherein the data compression scheme with be modified when the latency of data transmission exceeds a predetermined threshold.

[0081] Indeed, as explained above, the efficiency of the content-dependent compression/decompression schemes described herein is achieved, e.g., by virtue of the fact that the encoding tables are based on, and specifically designed for, the known data model. However, in situations where the data model is may be modified, the efficiency of the content-dependent scheme may be adversely affected, thereby possibly resulting in a reduction in compression efficiency and/or an increase in the overall latency of data transmission. In such a situation, as a backup system, the data compression controller can switch to a content-independent scheme that provides improved compression efficiency and reduction in latency as compared to the primary content-dependent scheme.

[0082] In yet another embodiment of the present invention, when the efficiency of a content-dependent scheme falls below a predetermined threshold based on, e.g., a change in the data structure of the data stream, the present invention preferably comprises an automatic mechanism to adaptively modify the encoding tables to generate optimal encoding tables (using the process described above with reference to **FIG. 3**).

[0083] **FIG. 5** is a detailed block diagram illustrates an exemplary content-independent data compression system **110** that may be employed herein. Details of this data compression system are provided in U.S. Pat. No. 6,195,024, which is fully incorporated herein by reference. In this embodiment, the data compression system **110** accepts data blocks from an input data stream and stores the input data block in an input buffer or cache **115**. It is to be understood that the system processes the input data stream in data blocks that may range in size from individual bits through complete files or collections of multiple files. Additionally, the input data block size may be fixed or variable. A counter **120** counts or otherwise enumerates the size of input data block in any convenient units including bits, bytes, words, and double words. It should be noted that the input buffer **115** and counter **120** are not required elements of the present invention. The input data buffer **115** may be provided for buffering the input data stream in order to output an uncompressed data stream in the event that, as discussed in further detail below, every encoder fails to achieve a level of compression that exceeds an a priori specified minimum compression ratio threshold.

[0084] Data compression is performed by an encoder module **125** that may comprise a set of encoders E1, E2, E3 . . . En. The encoder set E1, E2, E3 . . . En may include any number "n" (where n may=1) of those lossless encoding techniques currently well known within the art such as run length, Huffman, Lempel-Ziv Dictionary Compression, arithmetic coding, data compaction, and data null suppression. It is to be understood that the encoding techniques are selected based upon their ability to effectively encode different types of input data. It is to be appreciated that a full complement of encoders are preferably selected to provide a broad coverage of existing and future data types.

[0085] The encoder module **125** successively receives as input each of the buffered input data blocks (or unbuffered input data blocks from the counter module **120**). Data compression is performed by the encoder module **125** wherein each of the encoders E1 . . . En processes a given input data block and outputs a corresponding set of encoded data blocks. It is to be appreciated that the system affords a user the option to enable/disable any one or more of the encoders E1 . . . En prior to operation. As is understood by those skilled in the art, such feature allows the user to tailor the operation of the data compression system for specific applications. It is to be further appreciated that the encoding process may be performed either in parallel or sequentially. In particular, the encoders El through En of encoder module **125** may operate in parallel (i.e., simultaneously processing a given input data block by utilizing task multiplexing on a single central processor, via dedicated hardware, by executing on a plurality of processor or dedicated hardware systems, or any combination thereof). In addition, encoders E1 through En may operate sequentially on a given unbuffered or buffered input data block. This process is intended to eliminate the complexity and additional processing overhead associated with multiplexing concurrent encoding techniques on a single central processor and/or dedicated hardware, set of central processors and/or dedicated hardware, or any achievable combination. It is to be further appreciated that encoders of the identical type may be applied in parallel to enhance encoding speed. For instance, encoder E1 may comprise two parallel Huffman encoders for parallel processing of an input data block.

[0086] A buffer/counter module **130** is operatively connected to the encoder module **125** for buffering and counting the size of each of the encoded data blocks output from encoder module **125**. Specifically, the buffer/counter **130** comprises a plurality of buffer/counters BC1, BC2, BC3 . . . BCn, each operatively associated with a corresponding one of the encoders E1. En. A compression ratio module **135**, operatively connected to the output buffer/counter **130**, determines the compression ratio obtained for each of the enabled encoders E1. En by taking the ratio of the size of the input data block to the size of the output data block stored in the corresponding buffer/counters BC1 . . . BCn. In addition, the compression ratio module **135** compares each compression ratio with an a priori-specified compression ratio threshold limit to determine if at least one of the encoded data blocks output from the enabled encoders E1 . . . En achieves a compression that exceeds an a priori-specified threshold. As is understood by those skilled in the art, the threshold limit may be specified as any value inclusive of data expansion, no data compression or expansion, or any arbitrarily desired compression limit. A description module **138**, operatively coupled to the compression ratio module **135**, appends a corresponding compression type descriptor to each encoded data block which is selected for output so as to indicate the type of compression format of the encoded data block. A data compression type descriptor is defined as any recognizable data token or descriptor that indicates which data encoding technique has been applied to the data. It is to be understood that, since encoders of the identical type may be applied in parallel to enhance encoding speed (as discussed above), the data compression type descriptor identifies the corresponding encoding technique applied to the encoded data block, not necessarily the specific encoder. The encoded data block having the greatest compression ratio along with its corresponding data compression type descriptor is then output for subsequent data processing or transmittal. If there are no encoded data blocks having a compression ratio that exceeds the compression ratio threshold limit, then the original unencoded input data block is selected for output and a null data compression type descriptor is appended thereto. A null data compression type descriptor is defined as any recognizable data token or descriptor that indicates no data encoding has been applied to the input data block. Accordingly, the unencoded input data block with its corresponding null data compression type descriptor is then output for subsequent data processing or transmittal.

[0087] Again, it is to be understood that the embodiment of the data compression engine of **FIG. 5** is exemplary of a preferred compression system which may be implemented in the present invention, and that other compression systems and methods known to those skilled in the art may be employed for providing accelerated data transmission in accordance with the teachings herein. Indeed, in another embodiment of the compression system disclosed in the above-incorporated U.S. Pat. No. 6,195,024, a timer is included to measure the time elapsed during the encoding process against an a priori-specified time limit. When the time limit expires, only the data output from those encoders (in the encoder module **125**) that have completed the present encoding cycle are compared to determine the encoded data with the highest compression ratio. The time limit ensures that the real-time or pseudo real-time nature of the data encoding is preserved. In addition, the results from each

encoder in the encoder module **125** may be buffered to allow additional encoders to be sequentially applied to the output of the previous encoder, yielding a more optimal lossless data compression ratio. Such techniques are discussed in greater detail in the above-incorporated U.S. Pat. No. 6,195,024.

[**0088**] Referring now to **FIG. 6, a** detailed block diagram illustrates an exemplary decompression system that may be employed herein or accelerated data transmission as disclosed in the above-incorporated U.S. Pat. No. 6,195,024. In this embodiment, the data compression engine **180** accepts compressed data blocks received over a communication channel. The decompression system processes the input data stream in data blocks that may range in size from individual bits through complete files or collections of multiple files. Additionally, the input data block size may be fixed or variable.

[**0089**] The data decompression engine **180** comprises an input buffer **155** that receives as input an uncompressed or compressed data stream comprising one or more data blocks. The data blocks may range in size from individual bits through complete files or collections of multiple files. Additionally, the data block size may be fixed or variable. The input data buffer **55** is preferably included (not required) to provide storage of input data for various hardware implementations. A descriptor extraction module **160** receives the buffered (or unbuffered) input data block and then parses, lexically, syntactically, or otherwise analyzes the input data block using methods known by those skilled in the art to extract the data compression type descriptor associated with the data block. The data compression type descriptor may possess values corresponding to null (no encoding applied), a single applied encoding technique, or multiple encoding techniques applied in a specific or random order (in accordance with the data compression system embodiments and methods discussed above).

[**0090**] A decoder module **165** includes one or more decoders D1 . . . Dn for decoding the input data block using a decoder, set of decoders, or a sequential set of decoders corresponding to the extracted compression type descriptor. The decoders D1 . . . Dn may include those lossless encoding techniques currently well known within the art, including: run length, Huffman, Lempel-Ziv Dictionary Compression, arithmetic coding, data compaction, and data null suppression. Decoding techniques are selected based upon their ability to effectively decode the various different types of encoded input data generated by the data compression systems described above or originating from any other desired source.

[**0091**] As with the data compression systems discussed in the above-incorporated U.S. Pat. No. 6,195,024, the decoder module **165** may include multiple decoders of the same type applied in parallel so as to reduce the data decoding time. An output data buffer or cache **170** may be included for buffering the decoded data block output from the decoder module **165**. The output buffer **70** then provides data to the output data stream. It is to be appreciated by those skilled in the art that the data compression system **180** may also include an input data counter and output data counter operatively coupled to the input and output, respectively, of the decoder module **165**. In this manner, the compressed and corresponding decompressed data block may be counted to ensure that sufficient decompression is obtained for the input data block.

[**0092**] Again, it is to be understood that the embodiment of the data decompression system **180** of **FIG. 6** is exemplary of a preferred decompression system and method which may be implemented in the present invention, and that other data decompression systems and methods known to those skilled in the art may be employed for providing accelerated data transmission in accordance with the teachings herein.

[**0093**] It is to be appreciated that a data transmission acceleration system according to the present invention offers a business model by which market data vendors and users in the financial information services industry can receive various benefits. For example, the present invention affords transparent multiplication of bandwidth with minimal latency. Experiments have shown that increased bandwidth of up to 3 times can be achieved with minimal latency. Furthermore, proprietary hardware, including chip and board designs, as well as custom embedded and application software and algorithms associated with accelerated data transmission provide a cost-effective solution that can be seamlessly integrated with existing products and infrastructure. Moreover, the data acceleration through real-time compression and decompression affords a dramatic reduction in ongoing bandwidth costs. Further, the present invention provides mechanism to differentiate data feeds from other vendors via enriched content or quantity of the data feed. In addition, a data compression scheme according to the present invention provides dramatically more secure and encrypted feed from current levels, thus, providing the ability to employ a secure and accelerated virtual private network over the Internet for authorized subscribers or clients with proprietary hardware and software installed. Moreover, the present invention offers the ability to reduce a client's ongoing monthly bandwidth costs as an incentive to subscribe to a vendor's data feed service.

[**0094**] The present invention is readily extendable for use on a global computer network such as the Internet. This is significant since it creates a virtual private network and is important for the market data vendors and others due to its reduced cost in closed network/bandwidth solutions. In effect, the data vendors get to "ride for free" over the world's infrastructure, while still providing the same (and enhanced) services to their customers.

[**0095**] Although illustrative embodiments have been described herein with reference to the accompanying drawings, it is to be understood that the present invention is not limited to those precise embodiments, and that various other changes and modifications may be affected therein by one skilled in the art without departing from the scope or spirit of the invention. All such changes and modifications are intended to be included within the scope of the invention as defined by the appended claims.

What is claimed is:

1. A method for providing accelerated transmission of data over a network, comprising the steps of:

receiving a data stream for transmission over a communication channel;

compressing, in real-time, the data stream at a compression rate that increases the effective bandwidth of the communication channel;

transmitting the compressed data stream over the communication channel; and

decompressing, in real time, the compressed data stream received over the communication channel.

2. The method of claim 1, wherein the step of compressing comprises the steps of compressing the data stream using a statistical compression algorithm, wherein the statistical compression algorithm uses code tables that are derived based on a data model associated with the data stream.

3. The method of claim 2, wherein the statistical compression algorithm comprises Huffman encoding.

4. The method of claim 2, wherein the statistical compression algorithm comprises Arithmetic encoding.

5. The method of claim 2, wherein the code tables comprise a state machine.

6. The method of claim 5, wherein the state machine comprises one or more global states each having one or more code tables associated therewith.

7. A method for providing accelerated transmission of data over a network, comprising the steps of:

receiving a data stream for transmission over a communication channel;

compressing, in real-time, the data stream;

transmitting the compressed data stream over the communication channel; and

decompressing, in real time, the compressed data stream received over the communication channel, wherein a latency associated with the compressing, transmitting and decompression steps is less than a latency associated with transmitting the data stream in uncompressed format.

8. The method of claim 7, wherein the step of compressing comprises the steps of compressing the data stream using a statistical compression algorithm, wherein the statistical compression algorithm uses code tables that are derived based on a data model associated with the data stream.

9. The method of claim 8, wherein the statistical compression algorithm comprises Huffman encoding.

10. The method of claim 8, wherein the statistical compression algorithm comprises Arithmetic encoding.

11. The method of claim 8, wherein the code tables comprise a state machine.

12. The method of claim 8, wherein the state machine comprises one or more global states each having one or more code tables associated therewith.

13. A method for providing accelerated transmission of data over a network, comprising the steps of:

compressing a data stream using a content-dependent compression system that is constructed based on a data model associated with the data stream;

transmitting the compressed data stream over a communication channel; and

utilizing a content-independent compression system to compress the data stream, when a compression ratio obtained using the content-dependent system falls below a predetermined threshold, or when a latency associated with the data transmission exceeds a predetermined threshold.

14. The method of claim 13, wherein the step of compressing using a content-independent system comprises the steps of compressing the data stream using a statistical compression algorithm, wherein the statistical compression algorithm uses code tables that are derived based on a data model associated with the data stream.

15. The method of claim 14, wherein the statistical compression algorithm comprises Huffman encoding.

16. The method of claim 14, wherein the statistical compression algorithm comprises Arithmetic encoding.

17. The method of claim 14, wherein the code tables comprise a state machine.

18. The method of claim 17, wherein the state machine comprises one or more global states each having one or more code tables associated therewith.

* * * * *